Amendments to the Specification

Please replace [0004] with the following amended paragraph.

[0004] Application Serial No. 10/620,076 ~~Docket No.: QN1023.US~~, entitled **"METHOD AND SYSTEM FOR PROCESSING NETWORK DATA PACKETS";** and

Please replace [0005] with the following amended paragraph.

[0005] Application Serial No. 10/619,719 ~~Docket No. QN1024.US~~, entitled **"METHOD AND SYSTEM FOR PROCESSING NETWORK DATA PACKETS"**, the disclosure of which are incorporated herein by reference in their entirety.

Please replace [0078] with the following amended paragraph.

[0078] Figure 3J is a block diagram of an input verification processor ("IVP or IPV"), according to one aspect of the present invention;

Please replace [0095] with the following amended paragraph.

[0095] Figure 2A shows a typical storage area network 100 with host systems 102, 104, 107 and 108 coupled to various disks 103, 105, 106 and 109 (in network systems 102A, 104A, 107A and 108A) via IP network 101. The description of various adaptive aspects of the present invention below, are based on host 104, however, that is merely to illustrate one aspect of the present invention. Host system 104 (or others) are not described in detail, but it includes a central processing unit (CPU), a system memory (typically, random access memory "RAM"), read only memory (ROM) coupled to a system bus and a DMA controller unit.

Please replace [0098] with the following amended paragraph.

[0098] The present invention may be used on a PCI development board with a Field Programmable gate Array ("FPGA"). The chip may also be integrated into an Application Specific Integrated Circuit ("ASIC") with an embedded serialize/ de-

serializer ("SERDES" 203, Fig. 2B) and internal programmable RAM 207 and SDRAM 204.

Please replace [0099] with the following amended paragraph.

[0099] Figure 2B shows a top-level block diagram of system 200 using system 300 as described below in detail on an FPGA board. Figure 2B shows system 300 that includes an embedded processor 206 (which may include more than one processor) and a TCP/IP accelerator 202 that implements the TCP/IP protocol in hardware. GBIC 205 couples system 300 to a network.

Please replace [0101] with the following amended paragraph.

[0101] Figure 3A shows a block diagram of system 300 according to one aspect of the present invention, with various components described below. Outbound Processor ("OAP") 312, RISC Memory Interface (RMI) 313, Inbound Processor ("IAP") 307 and the Non-Data PDU FIFO block ("NPF") 314 implement the Upper Layer Protocol Processing (ULPP) Subsystem. The ULPP Subsystem, along with downloadable firmware, provides a mechanism for processing various protocols that can run on top of the TCP/IP protocol. iSCSI is one example of an upper level protocol that could be run by ULPP Subsystem.

Please replace [0103] with the following amended paragraph.

[0103] MAC Receive module 303, Inbound FIFO Block (IFB) 325, the IP Verify / Input Queuing module ("IPV") 302A, IP fragment Processor ("IFP") 305 and the Inbound TCP Processor ("ITP") 306 implement the Inbound TCP/IP Hardware Stack, which processes all inbound networking packets destined for Host 104 or the ULPP Subsystem.

Please replace [0105] with the following amended paragraph.

[0105] PCI/PCI-X Interface 341 (same as 201 and 201A, Figure 2B) and direct memory access ("DMA") Arbiter (DA) 342 implement a DMA subsystem that is used to transfer data between system 300 and host 104. Network Request Manager ("NRM") 333

and the Network Completion Manager ("NCM") 336 implement a subsystem for transferring messages between the TCP/IP hardware engines and host 104 via 341 and 342. SCSI Request Manager ("SRM") 334 and SCSI Completion Manager ("SCM") 335 perform the same function for the ULPP subsystem. Outbound DMA Engine ("ODE") 338 and Inbound DMA Engine ("IDE") 317 are used to transfer network data between Host 104 and System 300. This data can consist of TCP, IP or MAC level packet data. The remaining modules of [[sytem]]]] system 300 provide other support functions for the subsystems described below.

Please replace [0138] with the following amended paragraph.

[0138] The process of sending data includes signaling OIP 308 to build the IP and MAC layer headers, building the TCP header and then passing the header and data to OIP 308 to be sent onto the Ethernet link. It also includes saving [[Save]] new connection state in NCB and write the NCB to Local RAM 337 for later use when ACKs are returned from remote node via TTM 323 interface. As ACKs return, OTP 309 sends more data if needed or else finish processing request and passes a completion message to Host 104 signaling that the request is done.

Please replace [0147] with the following amended paragraph.

[0147] IP requests coming from Host 104 are in two forms, fully formed datagrams to be passed without modification or IP data to have a header attached to it. For fully formed datagrams, System 300 adds a MAC header and passes it to Outbound FIFO Block OFB 326. For IP data requests, OIP 308 builds complete IP header from entries in the NCB. OIP 308 may fragment the resulting datagram and add a MAC header. This means that all relevant IP fields in the NCB are filled before the send request is made.

Please replace [0151] with the following amended paragraph.

[0151] Ethernet MAC 304A (or MAC 304A) supports a full duplex operation and supports a connection to an external Serializer/Deserializer (SerDes) via a Ten Bit Interface (TBI). Ethernet MAC 304A handshakes received frame data for inbound FIFO

(IFB) 325 and verifies CRC. It then provides a signal to inbound FIFO (IFB) 325 to flush a current frame if a received frame is too short, too long, invalid EOP, invalid transmission character, or bad cyclic redundancy check ("CRC").

Please replace [0152] with the following amended paragraph.

[0152] Ethernet MAC 304A can source a status word to Inbound FIFO (IFB) 325 as the last word of each frame, which specifies frame length, broadcast, multicast, unicast and length of the MAC header and pad 2 bytes before MAC header to the packets intended for inbound FIFO (IFB) 325 to align the IP header on a 64 bit boundary. Ethernet MAC 304A also adjusts MAC header length and total length in status word to account for this.

Please replace [0153] with the following amended paragraph.

[0153] Ethernet MAC 304A generates CRC for transmit frames, supports reception of Ethernet and IEEE 802.3 frames, and supports VLAN/Priority for which a receiver removes VLAN tags, if present, to keep subsequent protocol headers aligned. The VLAN tag is passed up as part of the status word. Ethernet MAC 304A recognizes a pause packet and provides a pause signal to OIP 308; and supports 1- 4 MAC unicast addresses (reception). Ethernet MAC 304A also provides receive error counters, including CRC error, invalid transmission characters, loss of signal/loss of sync greater than a certain value, for example 10 ms, frame too short, or frame too long. It also provides counters for: transmitted frame count, transmitted byte count, received frame count, and received byte count.

Please replace [0154] with the following amended paragraph.

[0154] Ethernet MAC 304A also generates and checks parity, accepts all packets to multicast addresses, supports auto-negotiation as defined in IEEE802.3 Section 37, and inserts source MAC address in transmitted frame 303B.

Please replace [0175] with the following amended paragraph

[0175] TCP data passed to Host 104 has the TCP header stripped. "FIN" [[FIN]] segments as well as segments for unknown connections are passed to Host 104 with their headers. Details of ITP 306 are discussed below.

Please replace [0236] with the following amended paragraph

[0236] One field in the NCB, the TCP Timer Scale Factor will now be described in more detail. Each TCP timer in System 300 is referenced to a local timer and is defined as a certain number of local timer ticks. The scale factor ("SF") is used to adjust the time interval between timer ticks, on a per connection basis. This is done to allow for faster timeouts on connections that are on a very small network versus connections being run across a very large network. The scale factor is defined as a 3 bit field in the NCB and is an exponential multiplier. The timer tick interval is increased by a factor of 2^{SF}. The scale factor is used to increase or decrease the timer tick from that defined in the current BSD 4.4 release. A scale factor of 2 uses the same timer defined in the BSD implementation. Scale factors 1 and 0 divide the timers by 2 and 4 respectively. Scale factors of 3 or greater increase the timers by a power of two for each increment above 2.

Please replace [0254] with the following amended paragraph

[0254] To send a MAC frame, host 104 sends a descriptor to OIP 308. Thereafter, OIP 308 programs ODE 338 to move Ethernet frames from host memory to outbound FIFO OFB 326. The entire frame is placed into outbound FIFO OFB 326 before the frame is sent. Once a frame is sent, OIP 308 sends a completion message to host 104 through NCM 336. OIP 308 does not process data, but copies it into the outbound FIFO and then sends it. This means that Host 104 must create a complete Ethernet frame (or IEEE802.3 frame, if desired).

Please replace [0262] with the following amended paragraph

[0262] If the total datagram length fits into one IP packet, OIP 308 builds both the MAC and IP headers in outbound FIFO OFB 326. Destination MAC address is copied from the NCB and source MAC address is copied from the Ethernet MAC Address register. For an IP Header, the IP Version field is set to either 4 or 6, depending on the

"IPv6" bit in the NCB. IP header length field is calculated by adding 5 to the IP Option Length field from the NCB. IP Type of Service is copied from the NCB. IP packet length field is calculated from data length field in the descriptor plus the size of the IP header, w/ options. IP Identifier field is taken from a register maintained on System 300 that is incremented for each datagram. IP fragment flags and offset are all set to zero. IP Time to Live is copied from the NCB. IP Protocol field is copied from the NCB. IP Checksum is initially written as zero and is later rewritten after all the data has been moved and the checksum calculated.

Please replace [0263] with the following amended paragraph.

[0263] IP Source Address is copied from the port's IP Address register. IP Destination Address is copied from the NCB. If the IP Options bit is set, OIP 308 programs ODE [[317]] 338 to move the fully formed IP options data from Host 104 memory down to the outbound FIFO. After all the other header fields are filled in, OIP 308 sends the calculated checksum with a tag that tells the MAC to write it into the IP checksum field. Note that the IP checksum is always at a fixed offset from the beginning of the Ethernet frame. Once all the data is down in the FIFO, OIP 308 sends a completion message to Host 104 as described.

Please replace [0272] with the following amended paragraph.

[0272] Figure 4A shows how an initial network IOCB is read from host 104, and processed to transmit TCP data (404A-404G) . TCP data transmission also goes through a "Delayed Request" process (described later) to process ACK packets received for the data sent. OTP 309 reads a network IOCB and gets the NCB from local RAM 337. The NCB is moved to TTM 323. The network IOCB is linked to the NCB as a Delayed Request Block (DRB). OTP 309 verifies if a TCP window is open to send at least one segment. If not, an idle signal is sent to RA 310. If the TCP window is open, the first DRB linked to the NCB is fetched and OIP 308 is signaled to build MAC and IP headers in outbound FIFO 326.

Please replace [0273] with the following amended paragraph

[0273] After OIP 308 is done, OTP 309 builds its header in outbound FIFO OFB 326. Each field of the TCP header is filled in as follows: Source and Destination TCP ports are copied from the NCB. TCP sequence and acknowledgement numbers are copied from the NCB. TCP header length is calculated. TCP flags are copied from the NCB. Hardware sets the ACK flag regardless of the state of the flag in the NCB. TCP Window Size is copied from the current value in the NCB. TCP checksum is initially set to zero and then adjusted

Please replace [0274] with the following amended paragraph

[0274] OTP 309 processes NCB and adds a timestamp, if the connection is configured. OTP 309 sends TCP data from host memory to outbound FIFO OFB 326 via ODE 338. As the TCP header and data are passed to OIP 308, OIP 308 calculates the TCP checksum. If a retransmission timer is not already running on this connection, OTP 309 links the NCB on the timer queue for the retransmission timer. After the last word of data is passed to OIP 308, OIP 308 sends the calculated TCP checksum with a tag that tells OFB 326 to write it into the TCP checksum field, and the frame is sent. If all the data for the IOCB has been sent, OTP 309 writes the sequence number of the last byte of data for the IOCB in the DRB. OTP 309 also sets the Last Sequence number valid flag. Thereafter, OTP 309 updates all NCB entries and does a "write-back" of the NCB to local RAM 337.

Please replace [0281] with the following amended paragraph

[0281] OTP 309 reads the delayed request from local RAM 337 that is pointed to by the Snd_Max Descriptor Address field in the NCB. OTP 309 signals OIP 308 to build MAC and IP headers in outbound FIFO OFB 326. When OIP 308 is done, OTP 309 build's the header in the outbound FIFO OFB 326.

Please replace [0282] with the following amended paragraph

[0282] OTP 309 processes and adds a timestamp option, if connection is configured. OTP 309 sends TCP data from host memory to outbound FIFO OFB 326. As TCP

header and data are passed to OIP 308, OTP 309 calculates the TCP checksum. If a retransmission timer is not already running, OTP 309 links the NCB on the timer queue for retransmission timer.

Please replace [0297] with the following amended paragraph

[0297] The following process is used to send an ACK only packet:

TTM 323 signals RA 310 that it has an NCB that needs processing. RA 310 signals back to TTM 323 if it wins arbitration. TTM 323 reads the NCB from memory 337 and asserts a request to OTP 309.

OTP 309 checks action flags in the NCB. In this case, the SAN flag is set without the Window Update (WU) flag being set. OTP 309 signals OIP 308 to build MAC and IP headers in outbound FIFO OFB 326 as described above. When OIP 308 is done, OTP 309 builds its header in outbound FIFO OFB 326. The header indicates that only an ACK packet is being sent.

OTP 309 processes and adds a timestamp option, if connection is configured. As the TCP header is passed to OIP 308, OIP 308 calculates the TCP checksum. After the last word of data is passed down to OIP 308, OIP 308 sends the calculated TCP checksum with a tag that tells MAC 304 to write it into the TCP checksum field and send the frame. OTP 309 then sends an idle signal to RA 310; and the process ends.

Please replace [0302] with the following amended paragraph

[0302] OTP 309 checks for the SDA flag in NCB. OTP 309 signals OIP 308 to build MAC and IP headers in outbound FIFO OFB 326. When OIP 308 is done, OTP 309 builds its header in outbound FIFO OFB 326. This header indicates that only an ACK packet is being sent.

Please replace [0311] with the following amended paragraph.

[0311] OTP 309 signals OIP 308 to build MAC and IP headers in outbound FIFO OFB 326. When OIP [[326]] 308 is done, OTP 309 builds its header in outbound FIFO OFB 326. Source and Destination TCP ports are copied from the NCB.

Please replace [0312] with the following amended paragraph

[0312] OTP 309 processes and adds a timestamp option, if connection is configured. OTP 309 sends one byte of TCP data from host memory to outbound FIFO OFB 326. As TCP headers and data are passed to OTP 309, it calculates TCP checksum as well as counting the IP datagram length. After the last word of data is passed down to OIP 308, OIP 308 sends the calculated TCP checksum with a tag that tells MAC 304 to write it into the TCP checksum field and sends the frame. OTP 309 then sends an idle signal to RA 310, and the process ends.

Please replace [0321] with the following amended paragraph.

[0321] The t_idle timer is used in a TCP implementation to reset a congestion window on a connection that has been idle for a 'long' period, which may be one round trip delay. If no activity occurs on a connection for round trip time (RTT) [[RTT]], the congestion window value is reset back to one segment and a "slow start" begins when transmissions are restarted.

Please replace [0324] with the following amended paragraph

[0324] As frame packets (331) arrive from an Ethernet network, they are placed into inbound FIFO 325, while the MAC receiver (Rx, also refered to as MAC 303) 303 verifies the CRC. When the entire frame is in FIFO (IFB) 325 and if the CRC is valid, MAC 303 adds a "status word" to the beginning of the frame. The last word of data and the status word is written into FIFO (IFB) 325 with an "END" bit set. This status includes a frame length field, a header length field and status bits that indicate what type of address were matched to receive the frame. FIFO (IFB) 325 then signals IPV 302A that the frame is available. IPV 302A reads the frame out of FIFO (IFB) 325 and places it into Local RAM 337 using buflets acquired from BLM 302. IPV 302A links together as many

buflets as necessary to contain the entire frame. IPV 302A notes that the frame type field indicates that the frame is not destined for IP and send the frame to host 104 via IDE 317.

Please replace [0330] with the following amended paragraph.

[0330] When an IP frame arrives from an Ethernet network, it is placed into inbound FIFO (IFB) 325 while MAC receiver 303 verifies the CRC. When the entire frame is stored in FIFO (IFB) 325 and if the CRC is valid, MAC receiver 303 adds a status word to the frame. The last word of data and the status word is written into FIFO (IFB) 325 with an END bit set. This status includes a length field, a header length field and status bits that indicate what type of address was matched to receive the frame. FIFO (IFB) 325 then signals IPV 302A that the frame is available.

Please replace [0331] with the following amended paragraph

[0331] IPV 302A reads the frame out of FIFO (IFB) 325 and transfers it into Local RAM 337, using buflets acquired from Buflet List Manager 302. IPV 302A links together as many buflets as necessary to contain the entire frame. IPV 302A evaluates the frame type field, and if it indicates that the frame is destined for IP, then IPV 302A amends the first buflet data pointer to skip over the MAC header, based upon the MAC header length given in the status word. The IP header for the packet is placed into local RAM 337, and IPV 302A performs various validation checks, including IP header checksum and the comparison of the IP length against the actual received packet length.

Please replace [0336] with the following amended paragraph.

[0336] If the packet has not been otherwise disposed of and if IFP 305 is idle, IPV 302A passes the address for the first buflet of the packet and a copy of the IP header to IFP 305. If IFP 305 is not idle, the new packet is placed on the IFP 305 input list, and when IFP 305 is idle, IPV 302A re-reads the IP header and sends it. IFP 305 processes the fragmented datagram as described below and shown as 403 in Figure 4B.

Please replace [0339] with the following amended paragraph.

[0339] When the 1st fragment of a datagram is added to the reassembly list, the Nxt_Dgm_Lnk and the Prv_Dgm_Lnk are set to zero. If an entry already exists, the entry can be pointing to one or more datagrams (403B, 403C and 403E) that matched the hash. IFP 305 compares the IPSRC, IPDST, IPID and IPP fields (Figure 4B) of each datagram associated with the hash.

Please replace [0340] with the following amended paragraph.

[0340] If the datagram is not already on the list, it is added to the end of the list associated with the hash. When the 1st fragment of a datagram (403B) is added to the reassembly list, the Nxt_Dgm_Lnk and the Prv_Dgm_Lnk are set to the proper values.

Please replace [0342] with the following amended paragraph.

[0342] If the received fragment is not in-order, it is inserted in the ordered fragment list using the "Frg_Lnk" field (403D). The fragment offset in the IP header determines the insertion position on the list. If the fragment is placed before the fragment that was the first on the list for this datagram, the "Nxt_Dgm_Lnk" and "Prv_Dgm_Lnk" are copied into the buflet (403F).

Please replace [0343] with the following amended paragraph.

[0343] If the fragment is in-order with respect to another fragment then the buffers for the fragments is linked using the "Buf_Lnk" field (403H and 403G). TCP partial checksums are summed together and placed in the first buflet of the resulting list. When this linking takes place, the block also fixes the buflets if fragment overlap occurs. Further, if fragment overlap occurs, IFP 305 sets a 'C' bit for the datagram to force the TCP block to recalculate the TCP checksum, since the sum of the partial TCP checksums is invalid due to the overlap.

Please replace [0371] with the following amended paragraph.

[0371] After the checksum is validated TTM 323 is requested to fetch a NCB from local memory 337. Simultaneously, option block 329 searches for a time stamp. If a timestamp regarding the data is not found in the data received from IFP 305, and the data

header indicates that there may be a time stamp, then additional data is requested from MAM 301. Option block 329 then searches data in MAM 301 for timestamp and validation block [[330]] 330C verifies when a NCB was found by TTM 323.

Please replace [0390] with the following amended paragraph

[0390] Figure 3C7 shows the state machine process flow for data processor 306F. Data Processor 306F starts if ITP 306 determines that a segment should not be dropped or routed to the host. ITP 306 provides data processor (DP) [[DP]] 306F with miscellaneous header fields and in some cases the results of previously calculated values. DP 306F includes a Data Processor Controller (DC) whose functionality is described below:

Call Out of Order ("OOO") processor module (located within the data processor module 306F (Figure 3B of ITP 306) to trim data that doesn't fit in window.

Update buflet offset to a point past the header to the first byte of data in the TCP segment.

Set the fin flag in the buflet header if FIN flag in TCP header is set and trimming didn't trim from the end of the segment.

If segment is in order and nothing is on the Reassembly list

If NCB has delayed ack timer set on

Set NCB.SAN (send ack now).

Clear delay timer

Request Output Processor 306D to add to Output Request list.

Else

Start the delayed ack timer.

Queue NCB on timer list.

Update rcv_nxt.

Pass segment up to output processor 306D.

Else if Data is out of order or Reassembly list is not empty, pass segment to OOO Data Placement to place the buflet accordingly.

Call OOO processor to properly place data.

If in order data is received

Update rcv_nxt.

Set SAN (send ack now) bit.

Pass segment to output block.

Else (out of order data)

Set SDA (send duplicate ACK) bit.

Request Output Processor 306D to add to Output Request List

Signal Output Processor 306D if new in order data was received.

Update Re-assembly list if needed.

Please replace [0402] with the following amended paragraph.

[0402] Figure [[3D]] 3D2 is a block diagram of TTM 323 showing plural sub-modules that are used, according to one aspect of the present invention. TTM 323 includes plural registers (register set 323A) for ITP 306, IAP 307 and OAP 312, and provides read/write access for the foregoing modules. TTM 323 provides Fetch/Update/Flush functions for working registers at host memory or local RAM 337. TTM 323 also sends error signal(s) to ITP 306 and IAP 307 if a requested inbound NCB is not present in local RAM 337. TTM 323 also sends an overload signal to OTP 309 if local RAM 337 resources are not available.

Please replace [0403] with the following amended paragraph.

[0403] TTM 323 maintains timer functions for all TCP connections and co-ordinates all inbound and/or outbound channel access to network data structures. TTM 323 maintains a free list of data structures (323C) , delayed request blocks that are used to place IOCBs into a waiting FIFO for processing. DRBs are also used to place OAL associated with an IOCB into local RAM 337. When an OAL is placed into local RAM 337 it may be referred to as a delayed address list (DAL).

Please replace [0404] with the following amended paragraph.

[0404] TTM 323 also maintains a list of data structures to include NCBs (323D) for connections that need to be processed by OTP 309, and maintain an outbound request

list, which is a linked list of NCBs that are processed by OTP 309. Typically, ITP 306 and timer list manager 323E add NCBs to the list.

Please replace [0407] with the following amended paragraph.

[0407] TTM 323 processes an outbound IOCB and builds the local RAM 337 data structures for the outbound channel. For a new IOCB for data transfer, the entire IOCB and OALs are read and placed in local RAM 337 before data is actually sent. In order to build a data structure for a newly created TCP connection, OTP 309 requests TTM 323 to do the following (see also 399, Figure 3E):

Please replace [0408] with the following amended paragraph.

[0408] Read the new NCB from host memory (not shown) and place it in register set ("RS") 323A, and then write the new NCB into local RAM 337 using an entry from NCB free list. Thereafter, accept hash parameters into register set ("RS") 323A and generate a hash value. Link the new NCB (399B) off the hash table (399A, Fig 3E and 402E in Fig. 3F) using the generated hash value. This may involve following links from a hash table (399A, Fig. 3E) entry that has other connections that match the same hash value.

Please replace [0409] with the following amended paragraph.

[0409] For a hardware assisted TCP data transfer, the associated DRB is read from host memory and placed in register set ("RS") 323A. Resident DAL (399F) is linked to the most recent DRB (399D) using a DRB from the free list. It is noteworthy that many DALs (399F and 399G, Figure 3E) may be coupled to a single DRB. After all the DALs for the transfer have been linked, a resident DRB (399H or 399E) is linked to the NCB (399C) in local RAM 337 using a DRB from the free list.

Please replace [0411] with the following amended paragraph.

[0411] Read NCB from local RAM 337 using the address provided in the DRB and place the NCB in register set ("RS") 323A. For a hardware assisted TCP data transfer, read the DRB from host memory and place it in register set ("RS") 323A.

Thereafter, for each DAL, read the DAL from host memory and place it in register set 323A. Resident DAL is linked to the most recent DRB using a DRB from a free list. It is noteworthy that various DALs may be linked to a single DRB. After all DALs have been linked, link the resident DRB to the NCB in local RAM 337 using a DRB from the free list.

Please replace [0417] with the following amended paragraph.

[0417] Figure 3F shows a block diagram (402) of yet another aspect of the present invention showing the support provided by TTM 323 to manage an outbound request list. Typically, the outbound request list is used by ITP 306 and TTM 323 to signal OTP 309 that a NCB is ready for processing because of a timer event or change in credit values.

Please replace [0418] with the following amended paragraph.

[0418] TTM 323 manages a first in-first out ("FIFO") process requests to OTP 309 from ITP 306 or TTM 323 itself. The FIFO process results in an "outbound request list". The FIFO is implemented in RAM 337. The list may include requests to send linked list through NCBs (402C, 402F and 402G, Fig. 3F) in local ACKs, notifications of ACKs received and notifications of data packets timeouts.

Please replace [0419] with the following amended paragraph.

[0419] Typically, ITP 306 or TTM 323 place entries at the end of the request list (402B). If more than one request exists in the FIFO, TTM 323 requests arbitration to RA 310. When RA 310 grants permission, TTM 323 fetches the NCB. When the NCB is available, TTM 323 notifies OTP 309 and removes the outbound request from the head of the request list (402A).

Please replace [0421] with the following amended paragraph.

[0421] Figure 3G shows a block diagram (400) for re-assembly of inbound data structure. TTM 323 accepts hash parameters from hash table 399A and loads identified NCB so that ITP 306 has access to inbound re-assembly packet information. ITP 306

assumes that a NCB is located in local RAM 337. TTM 323 allows ITP 306 and IAP 307 to be active at the same time. ITP 306 provides local port, remote port, MA bits and remote IP address information from ITP 306 and determines the hash index values. TTM 323 uses the index value to see if an entry exists in hash table 399A. If an entry exists, the NCB is registered. If the value does not exist, the NCB is not registered.

Please replace [0422] with the following amended paragraph.

[0422] TTM 323 compares local port, remote port, MA bits and remote IP address information (400C, 400E and 400F, Fig. 3G) of the newly loaded NCB with the hash parameters in the hash parameter registers shown in register set 323A. If a match is found, ITP 306 is notified that a NCB is available. If a match is not found, TTM 323 searches for chained NCBs (400A, 400B, 400D, Fig. 3G), and if found, ITP 306 is notified.

Please replace [0425] with the following amended paragraph.

[0425] For OTP 309, TTM 323 maintains a "persist" timer 401C (with timer list lead 401A and timer list tail 401B) and a "retransmit timer" 401D for each connection. For ITP 306, TTM 323 maintains an idle timer (401F) and a delayed ACK timer (401E) for each connection.

Please replace [0426] with the following amended paragraph.

[0426] If a TCP connection needs to be timed and is not already on the timer list, OTP 309 or ITP 306 requests TTM 323 to add the connection's NCB to the timer list. When an NCB is added to the timer list, it is resident in TTM 323. When TTM 323 processes the timer list 401, it loads timer fields (401G) into a TTM 323 cache (not shown). However, timer link field may be resident in local RAM 337.

Please replace [0427] with the following amended paragraph.

[0427] At a pre-determined interval or programmable time, TTM 323 scans timer list 401 and checks for timer flags (401G). If a flag is set, TTM 323 compares the timer

value to the current tcp_now time. If the values are substantially equal, then the timer has elapsed.

Please replace [0428] with the following amended paragraph.

[0428] If a persist timer (401C) elapses, TTM 323 places an NCB to the outbound request list. If a re-transmit timer has elapsed, TTM 323 places the NCB on an outbound request list to re-transmit the oldest unacknowledged segment. If the delayed ACK timer has elapsed, TTM 323 places the NCB on outbound request list to have OTP 309 send an ACK.

Please replace [0433] with the following amended paragraph.

[0433] The following provides a list of various command functions that command processor "CP" 323B executes:

Please replace [0479] with the following amended paragraph.

[0479] Figure 3I show a block diagram, of OTP 309. The various modules (including module 309H and register block 309F) in OTP 309 access TTM 323 through main block 309C. The functionality of the plural components is shown below:

Please replace [0481] with the following amended paragraph.

[0481] Request Manager 309G: Downloads an IOCB, saves it in TTM 323, and determines what action needs to be taken. If this is an update or flush command, it passes the command to TTM 323 and exits. If data is included in the IOCB it has Outbound DMA Interface (also referred to as ODE Interface Module) 309A fetch the OAL chain and links them. After the NCB has all the information in it, control is passed to the Main Control block 309C to continue data processing. After Main Control block 309C is finished, request manager 309G saves the NCB.

Please replace [0487] with the following amended paragraph.

[0487] IPV 302A has been described above with respect to various other modules. The following describes various sub-modules of IPV 302A with respect to Figure 3J. IPV

302A includes registers 302A2 and an input processor 302A1 that is coupled to MAM 301 and BLM 302. Input processor 302A1 receives input data from IFB 325, processes the data and sends it to output processor 302A3, which is coupled, to IFP 305. Input processor 302A1 is also linked with IDE 317 and ILM 324.

Please replace [0489] with the following amended paragraph.

[0489] If IFB 325 has data to send to local RAM 337:

ifb_ipv_dav is asserted (FIFO has a complete frame to pass to memory) and blm_ipv_bav is asserted to start accepting frame.

dak and store first dword of data (status)

bak buflet from BLM 302. Save current buflet index and index of first buflet of frame, "frame head buflet."

1. Transfer frame to MAM 301.
   a) Send MAM 301 length strobe with length = minimum data [i.e. data remaining in FIFO 325 for the frame and space remaining in the buflet]
   b) Send MAM 301 address strobe with address = cur_buf_adr + 24
   c) Transfer data from FIFO (IFB) 325 to MAM 301 until either:
      i. The end bit from FIFO (IFB) 325 is set (indicating the end of the frame)
      ii. The number of bytes transferred in MAM 301 transaction = space remaining in the buflet (data_len = bufsize, indicating that the system should request another buflet.)
2. Check data in transit from FIFO (IFB)325 to MAM 301:

   Check type field (use header length in status word) to see if the frame is meant for IP

   If for IP:

   Verify IP header length (>=5 words = 20 bytes), header checksum, actual data length, ip version and discard packet and free buflets on any failure.

   Set buflet offset to skip over MAC header

Calculate TCP/UDP checksum, including pseudo header for frames where "More Fragments" bit is not set.

3. Update data structures:
   a) If not the first buflet (frame head buflet), update the control fields of the current buflet , including updating the buf_lnk field to point to the next buflet if frame continues. This requires baking the next buflet.

buf_lnk=cur_buf_ind or NULL if last buflet of frame.

buffer_offset = 24

signature = signature

buflet_data_len = save_data_len

All other fields in buflet control are null.

   b) Update loc_buf_lnk_tail to point to the current buflet
   c) Increment the buflet_cnt
4. Loop control decision
   if 2ci) above was true, then go to step 6.

   if 2cii) above was true then there is more data for this frame in FIFO (IFB) 325.

Next buflet received from BLM 302 in step 4.

Repeat steps 2-5 with appropriate addresses and lengths until all the data for the frame has been transferred.

5. Frame verified and stored in local RAM 337
   a) Update control fields of frame head buflet

buf_lnk=loc_buf_lnk

buf_lnk_tail=loc_buf_lnk_tail_ind

buffer_offset = if MAC: 26 ; if IP: 24 + mac_hdr_len

(26 because 24 for buflet control + 2 bytes of alignment padding on mac header.

(Mac_hdr_len includes this 2 bytes)

signature = signature

buflet_count = buf_cnt

buflet_data_len = For a frame where this is the ONLY buflet: save_data_len for a frame with multiple buflets: bufsize

tcp_checksum = tcp_checksum

length = if MAC: frame_length – 2 (align padding). if IP: IPLEN – (IPHL * 4)

I=1'b0 B=B, M=M, MA=MA, opcode= MAC || IP

6. Forward processing:

If not IP: Pass address of the frame head buflet to IDE 317 for forwarding to Host 104.

If IP and not known address: Pass address of frame head buflet to IDE 317 for forwarding to Host.

If IP and know address: add frame to tail of IFP 305 input list.

```
if !(empty) {
  lock_tail = 1
  store tail_frm_lnk->frm_lnk = frm_hd_buf_ind
  new_elem_ind = frm_bd_buf_ind
  add_to_list = 1
  lock_tail = 0
}
else {
  new_elem_ind = frm_hd_buf_ind
  add_to_list = 1
}
```

7. Error Processing:

a) Continue normal processing and storing frame to MAM 301 until the MAC header, IP header and 8 bytes of the IP payload are stored.

b) Continue to accept data out of FIFO (IFB) 325, but do not store to MAM 301 until end flag is set.

c) Write back current buflet control field as normal.

d) Write back frame head buflet control field as normal but with the following error values:

buf_lnk=16'b0

buf_lnk_tail=frm_hd_buf_ind (points to self)

buffer_offset = 26 (skip buflet control fields and mac align padding only)

signature = signature

buflet_count = 1

buflet_data_len = 24+ mac_hdr_len + IPHL*4 + 8

tcp_checksum = 16'b0

length = mac_hdr_len –2 + IPHL*4 + 8

I=1'b1, B=B, M=M, MA=MA, opcode=IP

e) Send address of frame head buflet to IDE 317.

f) Free rest of buflets through BLM 302 (loc_buf_lnk should still be pointing to the head of this list, and the last buflet has been linked on this list as usual through 'c' above.

Please replace [0493] with the following amended paragraph.

[0493] If it is an IP packet, OIP 308 requests TTM 323 to fetch the NCB from the host. After processing all the OALs in the IOCB, OIP 308 uses ODE 338 to fetch the OAL List associated with the IOCB and passes it to TTM 323. OIP 308 also reads the appropriate fields within the local NCB to build the IP and MAC Headers and writes these headers to Outbound FIFO _OFB 326.

Please replace [0494] with the following amended paragraph.

[0494] For source MAC address field, OIP 308 sends the index in the first byte to FIFO OFB 326. The MAC block converts this to a proper address. The index is taken from the first word of the NCB. The location of the source address in FIFO OFB 326 is maintained by padding it with zeros.

Please replace [0495] with the following amended paragraph.

[0495] For IP packets, OIP 308 also calculates the IP header checksum and TCP checksum of the data as it passes through flags when the locations of the IP and TCP checksum fields are being passed to Outbound FIFO OFB 326; reads Address/Length pairs from TTM 323 and pass them to ODE 338 to fetch packet data.

Please replace [0496] with the following amended paragraph.

[0496] OIP 308 handshakes data from ODE 338 and passes it to Outbound FIFO OFB 326 and byte packs data obtained from ODE 338/OTP 309. For an IP packet, OIP 308 also fragments data if the length from ODE 338 is greater than max_frame_size number of bytes (default 1500). This requires generation of a new header for each fragment.

Please replace [0500] with the following amended paragraph.

[0500] When all the data has been sent, OIP 308 passes the IP packet and TCP checksums to Outbound FIFO OFB 326 with a flag, which indicates it is the actual checksum data inserted in the packet. The last word of data has an end bit set on it along with the length.

Please replace [0503] with the following amended paragraph.

[0503] Outbound FIFO interface 308A handles all handshaking in the outbound pipeline through a byte packer and calculates IP header checksum and TCP/UDP checksum. Outbound FIFO interface 308A uses dav/dak signals to transfer data. It calculates the IP header checksum and TCP checksum of the data as it passes through. It

flags when the locations of the IP and TCP checksum fields are being passed to Outbound FIFO OFB 326; handshakes data from ODE 338, OTP 309, TTM Interface 308B and parses it to Outbound FIFO OFB 326; and byte packs data obtained from ODE 338/OTP309/TTMI 308B.

Please replace [0504] with the following amended paragraph.

[0504] When all the data has been sent, interface 308A passes the IP and TCP checksums to Outbound FIFO OFB 326 with a flag that indicates it is the actual checksum data to be inserted in the packet.

Please replace [0505] with the following amended paragraph.

[0505] TTM interface 308B reads outbound IP IOCB from RA 310 and pass to TTM 323 for temporary storage and saves the H (mac_hdr_only), U (UDP_En), Opcode_Embedded, and D (Disable_Comp) bits in transit. Interface 308B requests TTM 323 to fetch the NCB from the host or EP based upon the Opcode_Embedded bit above. After processing all the OALs in the IOCB, it uses ODE 338 to fetch bytes of the OAL associated with the IOCB and passes it to TTM 323 for temporary storage. Interface 308B reads the appropriate fields within the local NCB to build the IP and MAC Headers and writes these headers to Outbound FIFO OFB 326.

Please replace [0508] with the following amended paragraph.

[0508] Figure 3L shows a block diagram of IFP 305. The various aspects of IFP 305 with its sub-modules including local RAM interface 305B and control registers 305E will now be described. Figure 3L1 shows a link list data flow diagram for IP reassembly as performed by IFP 305.

Please replace [0511] with the following amended paragraph.

[0511] Input Processor 305D handshakes received IP packet headers from IPV 302A. If the received IP packet from IPV 302A is a complete datagram, processor 305D passes the received IP packet header and buflet pointer to output processor 305C.